Scientific and Educational Reports of the Faculty
of Science and Technology, Kochi University
Vol. 6 (2023), No. 1

同種写像を用いたディフィヘルマン鍵交換と IDベース暗号の実装

Implementations of Diffie-Hellman Key Exchange and ID-Based Encryption by Using Isogenies

境野 圭 * ,　塩田研一 †

Kei Sakaino*　Ken-ichi Shiota†

要旨

現在の情報通信はRSA暗号や楕円曲線暗号等の公開鍵暗号に支えられている。しかし，量子計算機が実用化に至ればこれらの安全性が脅かされることから，耐量子計算機暗号がいくつか提案されている。本研究では，その中でも楕円曲線間の同種写像を用いた暗号方式であるディフィヘルマン鍵交換（SIDH）とIDベース暗号（SIIBE）の実装を行い，実用上問題ない処理時間で動作することを確認した。論文では，同種写像を高速に計算する手法と，その最適なパラメータを解析した結果を述べた後，SIDH鍵交換とSIIBEの実行時間計測結果について報告する。

1　同種写像

$\mathbb{F}_q$ を標数が 5 以上の位数 q の有限体，$E : y^2 = x^3 + ax + b \ /\mathbb{F}_q$ を $\mathbb{F}_q$ 上の楕円曲線，$\mathcal{O}$ をその無限遠点とし，楕円曲線 $E/\mathbb{F}_q$ 上の $\mathbb{F}_q$-有理点の集合を $E(\mathbb{F}_q)$ と表す。また，自然数 m に対して，E の m-等分点の成す部分群を $E[m]$ と表す。定義体の標数 p について，$E[p] = \{\mathcal{O}\}$ であるような楕円曲線を超特異楕円曲線という。

$E' : Y^2 = X^3 + a'X + b'$ をもう一つの $\mathbb{F}_q$ 上の楕円曲線とし，$\mathcal{O}'$ をその無限遠点とする。有理写像 $\phi\colon E \to E'/\mathbb{F}_q$ が $\phi(\mathcal{O}) = \mathcal{O}'$ を満たすとき，ϕ を同種写像という。同種写像は自動的に加法準同型になり，その核 $\mathrm{Ker}(\phi) = \{\, P \in E \mid \phi(P) = \mathcal{O}' \,\}$ は E の部分群となる。

逆に，E の有限部分群 C に対して剰余群 E/C も楕円曲線の構造を持ち，自然な準同型

$$\phi\colon E \longrightarrow E/C$$

*高知大学大学院総合人間自然科学研究科理工学専攻情報科学コース
Information Science Course, Gaduate School of Science and Technology, Kochi University
†高知大学理工学部情報科学科
Department of Information Science, Faculty of Science and Technology, Kochi University

は C を核とする同種写像になる。ϕ 及び $E' = E/C$ の定義式を求めるには 次の Vélu の公式 [4] を用いる。

1.1 Vélu の公式

楕円曲線 $E : y^2 = x^3 + ax + b$ と，その有限部分群 C が与えられているとする。
$E' = E/C$: $Y^2 = X^3 + a'X + b'$ と同種写像

$$\begin{array}{ccc} \phi : E & \longrightarrow & E' \\ \cup & & \cup \\ (x, y) & \longmapsto & (X, Y) \end{array}$$

は次の式で得られる。

$Q = (x_Q, y_Q) \neq \mathcal{O} \in C$ に対し

$$g_Q^x := 3x_Q^2 + a, \quad g_Q^y := -2y_Q$$

とおいて

$$t_Q := \begin{cases} g_Q^x & \text{if} \quad Q \in E[2] \\ 2g_Q^x & \text{otherwise} \end{cases}, \quad u_Q := (g_Q^y)^2$$

とする。また，$S := (C - \{\mathcal{O}\})/\{\pm 1\}$ とし，

$$t := \sum_{Q \in S} t_Q, \quad u := \sum_{Q \in S} (u_Q + x_Q t_Q)$$

とおいて

$$a' := a - 5t, \quad b' := b - 7u$$

とする。X, Y は次の式で得られる。

$$X = x + \sum_{Q \in S} \left(\frac{t_Q}{x - x_Q} + \frac{u_Q}{(x - x_Q)^2} \right),$$

$$Y = y - \sum_{Q \in S} \left(\frac{2u_Q y}{(x - x_Q)^3} + \frac{t_Q(y - y_Q) - g_Q^x g_Q^y}{(x - x_Q)^2} \right).$$

1.2 ℓ^e -同種写像計算

Vélu の公式は核の位数オーダーの計算回数を必要とするが，暗号の設計では核の位数が数百ビットである必要があり，効率が悪い。そこで，同種写像を暗号に利用するためには，位数が小さい素数 ℓ のべき乗 ℓ^e であるような点 P_0 の生成する巡回部分群 $\langle P_0 \rangle$ を核として設定し，次のように e 回のステップに分けて同種写像を計算する [2, §4.2.2] 。

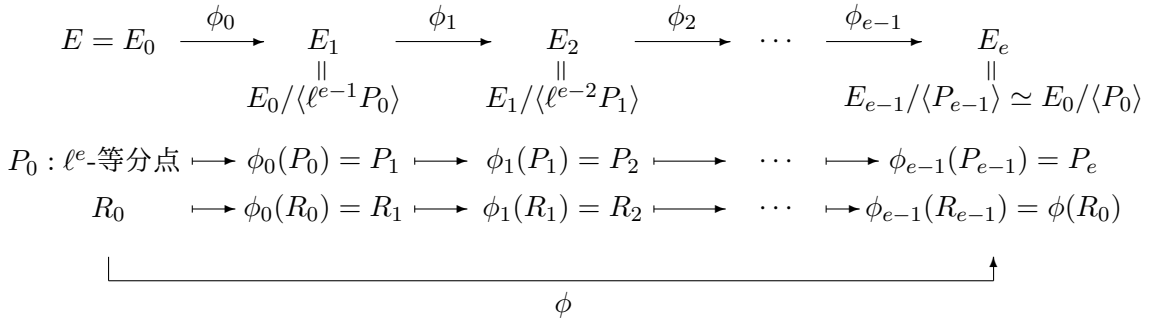

図 1: ℓ^e - 同種写像計算

このように，楕円曲線上の ℓ 倍算と，核の位数が ℓ の場合の Vélu の公式を複数回組み合わせることにより，核の位数が数百ビットであるときも同種写像計算が計算可能になる。

2 同種写像暗号

2.1 SIDH 鍵交換

同種写像を用いたディフィヘルマン鍵交換を SIDH (Supersingular Isogeny Diffie-Hellman) と呼ぶ [2, §3]。

Alice と Bob が鍵交換したいとする。事前準備として，小さな素数 ℓ_A, ℓ_B と，小さな整数 $f > 0$ を選び，素数 $p = f \times \ell_A^{e_A} \times \ell_B^{e_B} - 1$ を作成し公開する。曲線は

$$E(\mathbb{F}_{p^2}) \simeq (\mathbb{Z}/(1+p)\mathbb{Z})^2 \supseteq (\mathbb{Z}/\ell_A^{e_A}\mathbb{Z})^2 \oplus (\mathbb{Z}/\ell_B^{e_B}\mathbb{Z})^2$$

となる超特異楕円曲線 $E/\mathbb{F}_{p^2}$ を選び，位数が $\ell_A^{e_A}$ である点 $P_A, Q_A \in E[\ell_A^{e_A}]$，位数が $\ell_B^{e_B}$ である点 $P_B, Q_B \in E[\ell_B^{e_B}]$ を公開する。Alice と Bob はそれぞれ次の計算を行う。

Alice

- 点 $R_A := m_A P_A + n_A Q_A \in E[\ell_A^{e_A}]$ が位数 $\ell_A^{e_A}$ をもつようにランダムな整数 m_A, n_A ($0 < m_A, n_A < \ell_A^{e_A}$) を選び，R_A と m_A, n_A を秘匿する。
- $\langle R_A \rangle$ を核とする同種写像

  $$E \xrightarrow{\varphi} E_A := E/\langle R_A \rangle$$

  と $\varphi(P_B), \varphi(Q_B)$ を計算し，$E_A, \varphi(P_B), \varphi(Q_B)$ を公開する。

Bob

- 点 $R_B := m_B P_B + n_B Q_B \in E[\ell_B^{e_B}]$ が位数 $\ell_B^{e_B}$ をもつようにランダムな整数 m_B, n_B ($0 < m_B, n_B < \ell_B^{e_B}$) を選び，R_B と m_B, n_B を秘匿する。
- $\langle R_B \rangle$ を核とする同種写像

  $$E \xrightarrow{\psi} E_B := E/\langle R_B \rangle$$

  と $\psi(P_A), \psi(Q_A)$ を計算し，$E_B, \psi(P_A), \psi(Q_A)$ を公開する。

Alice

E_B と $\psi(P_A), \psi(Q_A)$ を用いて

$$m_A\psi(P_A) + n_A\psi(Q_A) = \psi(m_AP_A + n_AQ_A) = \psi(R_A)$$

を計算し，$\langle\psi(R_A)\rangle$ を核とする同種写像

$$E_B \quad \xrightarrow{\varphi'} \quad E_{AB} := E_B/\langle\psi(R_A)\rangle$$

を計算する。

Bob

E_A と $\varphi(P_B), \varphi(Q_B)$ を用いて

$$m_B\varphi(P_B) + n_B\varphi(Q_B) = \varphi(m_BP_B + n_BQ_B) = \varphi(R_B)$$

を計算し，$\langle\varphi(R_B)\rangle$ を核とする同種写像

$$E_A \quad \xrightarrow{\psi'} \quad E_{BA} := E_A/\langle\varphi(R_B)\rangle$$

を計算する。

このとき E_{AB} と E_{BA} は同型となるので，その j-不変量を共通鍵とする。その理由は次の通りである。

$$\begin{aligned} E_{AB} &= E_B/\langle\psi(R_A)\rangle \\ &\simeq E/\langle R_A, R_B\rangle \\ &\simeq E_A/\langle\varphi(R_B)\rangle \\ &= E_{BA} \end{aligned}$$

SIDH 鍵交換の流れを図 2 に示す。

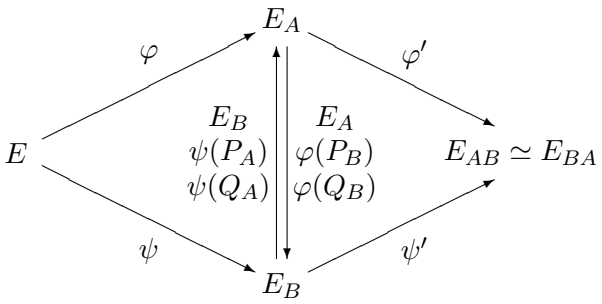

図 2: SIDH 鍵交換の流れ

注意 2.1.1. Alice か Bob のどちらかの強度が弱まらないように，$\ell_A^{e_A}$ と $\ell_B^{e_B}$ のビット数は同程度に設定する必要がある。また，f は p が 素数になるよう設定するが，強度のためには小さい整数であることが望ましい。

2.2 SIIBE

ID ベース暗号を，同種写像とペアリングを用いて実現することが可能である。これを SIIBE (Supersingular Isogeny Identity-Based Encryption) と呼ぶ [3]。

事前準備

鍵生成局が小さな素数 ℓ と，整数 $f > 0$ を選び，

- 素数 $p = f \times \ell^e - 1$
- 位数が $1 + p = f \times \ell^e$ である超特異楕円曲線 $E_0/\mathbb{F}_p$
- 核の位数が ℓ^e である同種写像 $\phi\colon E_0 \to E_1$

を決める。このとき $m = 1 + p$, $q = p^2$ として $E_0[m] = E_0(\mathbb{F}_q)$, $E_1[m] = E_1(\mathbb{F}_q)$ が成り立つ。E_0, E_1 は公開し，ϕ はマスタ秘密鍵として保持する。さらに

- ペアリング $e_0\colon E_0[m] \times E_0[m] \to \mathbb{F}_q^\times$, $e_1\colon E_1[m] \times E_1[m] \to \mathbb{F}_q^\times$
- $\mathbb{F}_q$ を ID 空間とするハッシュ関数 $H\colon \mathbb{F}_q \to E_0[m]$
- $\hat{g_0} \in E_0[m]$, $\hat{g_1} := \phi(\hat{g_0}) \in E_1[m]$

を決めて公開する。

秘密鍵の生成

鍵生成局が各ユーザ X の $\mathrm{ID}_X \in \mathbb{F}_q$ から暗号化鍵 $h_{0X} = H(\mathrm{ID}_X) \in E_0[m]$ を作り，$h_{1X} = \phi(h_{0X})$ を X へ復号鍵として渡す。

暗号化

ユーザ A に対して平文 M を暗号化するには乱数 $\zeta \in \mathbb{Z}/m\mathbb{Z}$ を選んで $c := \zeta\hat{g_1}$, $z := (e_0(h_{0A}, \zeta\hat{g_0}))^{\ell^e}$, $c_T := Mz$ を計算し

$$\mathrm{Enc}(M) := (c, c_T)$$

を暗号文とする。

復号

ユーザ A は暗号文 $C := (c, c_T)$ を受け取り

$$\mathrm{Dec}(c, c_T) := c_T(e_1(h_{1A}, c))^{-1}$$

を計算する。

正しく復号できる理由は次の通りである。

$$\begin{aligned}
\mathrm{Dec}(\mathrm{Enc}(M)) &= \mathrm{Dec}((c, c_T)) \\
&= c_T(e_1(h_{1A}, c))^{-1} \\
&= Mz(e_1(h_{1A}, \zeta\hat{g_1}))^{-1} \\
&= Mz(e_1(\phi(h_{0A}), \phi(\zeta\hat{g_0})))^{-1} \\
&= Mz((e_0(h_{0A}, \zeta\hat{g_0}))^{\ell^e})^{-1} = Mz\ z^{-1} = M.
\end{aligned}$$

ここで同種写像とペアリングの適応性（compatibility）

$$e_1(\phi(P),\phi(Q)) = e_0(P,Q)^{\deg(\phi)}$$

を用いた [1, §3]。

注意 2.2.1. z は 1 の f 乗根であるので，全探索による攻撃を避けるためには f を大きく設定しなければならない。

3 ℓ^e -同種写像計算の計算量削減

1.2 節で述べた ℓ^e -同種写像計算中の計算回数を削減させることで高速に同種写像を計算する手法を述べる。

まず，最も素朴に計算する方法の例を示す。$e = 11$，核を $P_0 \in E_0[\ell^{11}]$ としたとき，$\phi(R_0)$ を最も素朴に求める例は図 3 である。

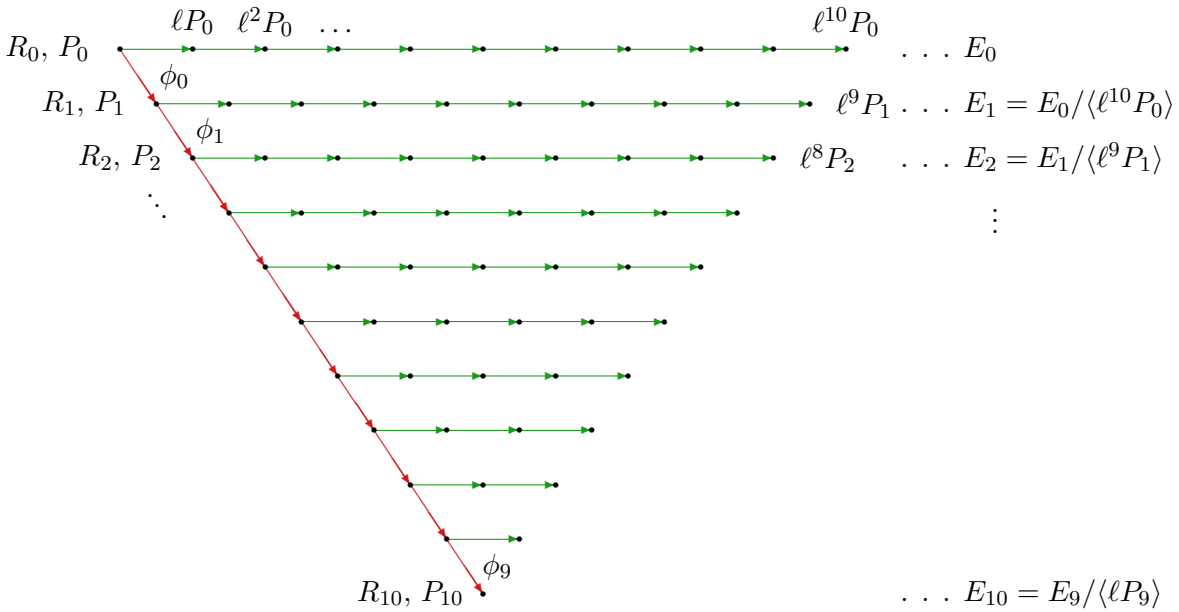

図 3: ℓ^{11}-同種写像の例

図 3 の有向グラフは，ℓ 倍算を緑の枝，同種写像計算を赤の枝で示し，ℓ^e-同種写像を計算するのに必要な計算を示している。計算手順は次のとおりである。楕円曲線 E_i 上の有理点 P_i を順次 ℓ 倍計算することで，$\ell^{e-i-1}P_i$ を求める。赤の枝はそれを核として，$\phi_i(R_i)$ と $\phi_i(P_i)$ を Vélu の公式で計算する。これらは E_{i+1} 上の点である。i を 0 から $e-1$ までインクリメントすることにより，目的となる楕円曲線 E_e 上の点 $\phi_{e-1}(R_{e-1}) = \phi(R_0)$ を得ることができる。ℓ 倍算と同種写像の計算回数の合計は $O(e^2)$ である。本手法は最も素朴な手法であるため simple method と呼ぶことにする。

この計算量は，次に示す有向グラフとすることにより $O(e \log e)$ まで削減することができる。この有向グラフを optimal tree と呼び，このときの手法を optimal method と呼ぶことにする。

$e = 16$ のときに optimal tree を生成する手法を図 4 から図 7 に示す。

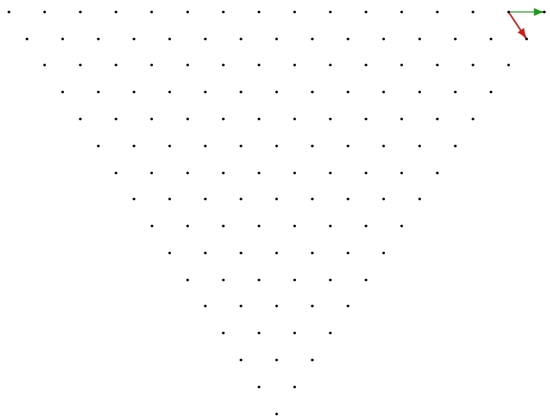

図 4: Optimal tree の生成 1

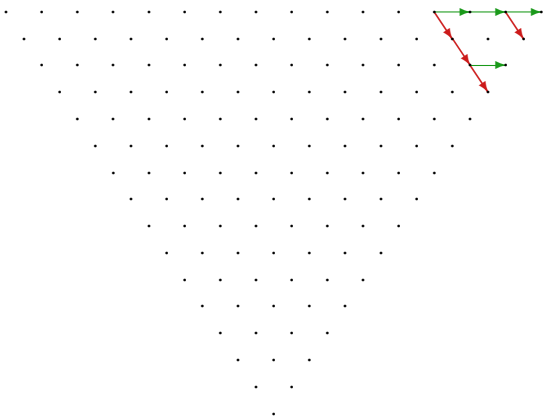

図 5: Optimal tree の生成 2

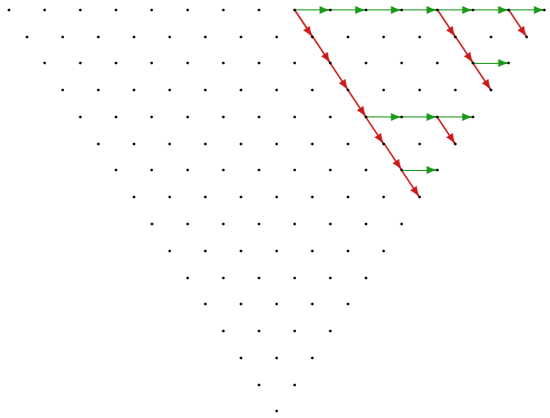

図 6: Optimal tree の生成 3

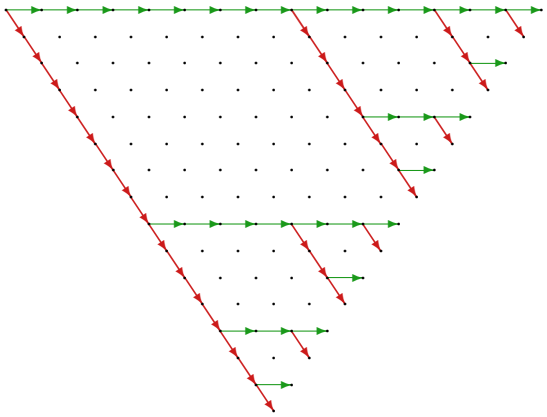

図 7: Optimal tree の生成 4

まず，初期設定として図 4 のように枝が 2 本の木を用意する。そして，図 4 と同じ木を 2 つ用意し，それらの根に繋がるように 4 本の枝を繋ぎ，図 5 を得る。次に，図 5 と同じ木を 2 つ用意し，それらの根に繋がるように 8 本の枝を繋ぎ，図 6 を得る。最後は，図 6 と同じ木を 2 つ用意し，それらの根に繋がるように 16 本の枝を繋ぎ，図 7 を得る。この図 7 が optimal tree である。

Optimal tree の枝の合計本数は，$e = 2^n$ のとき $2^n \times n = e \times \log e$ となる。従って，本手法を用いた場合の計算回数のオーダーは $O(e \log e)$ である。

e が $e = 2^n$ と表せないときは，左右のバランスを崩すことにより optimal tree を作成することが可能である。$e = 20, 29$ のときの例を図 8，9 に示す。

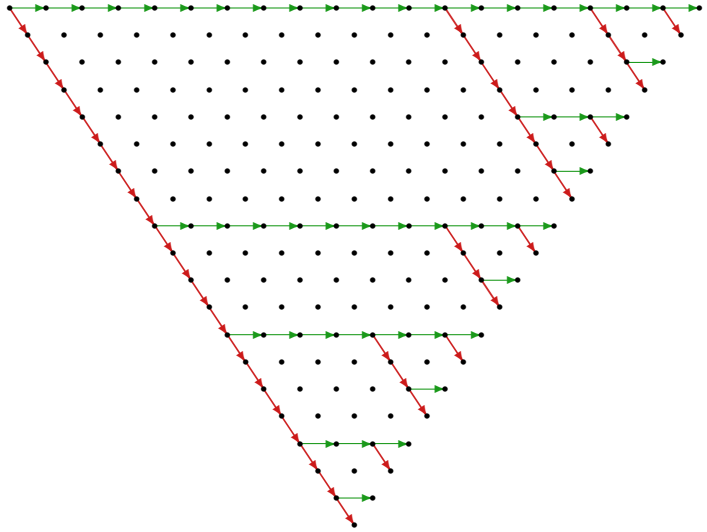

図 8: Optimal tree $e = 20$ のときの例

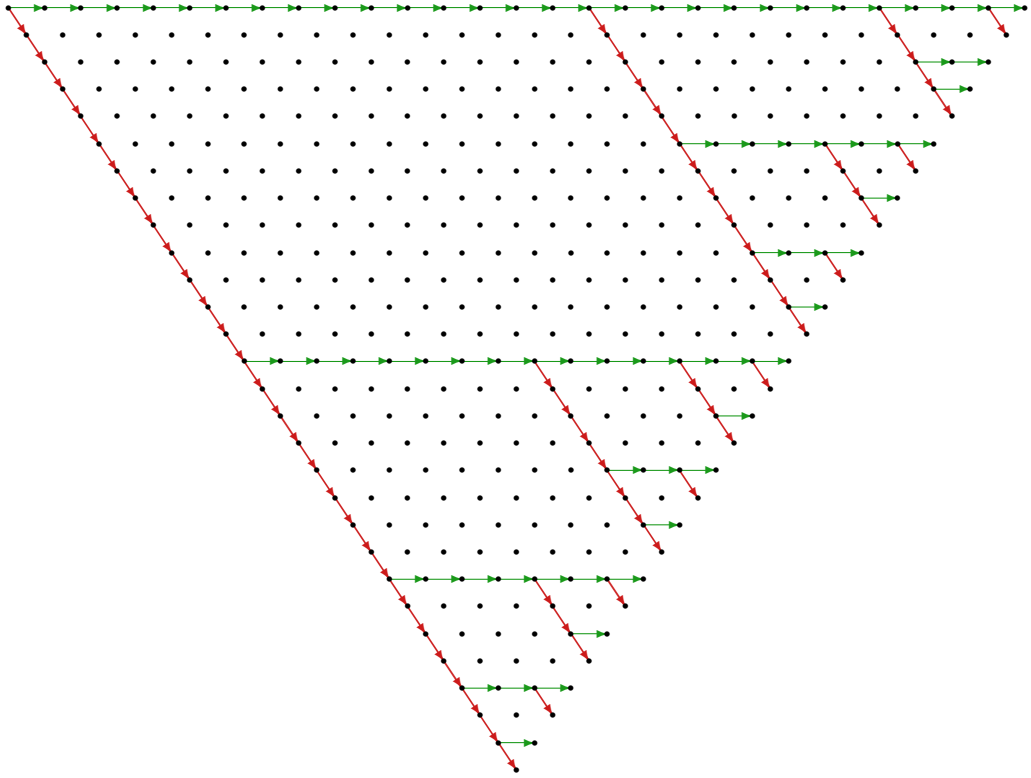

図 9: Optimal tree $e = 29$ のときの例

また，ℓ 倍算，同種写像計算の計算時間は少なからず計算機に依存する。実用の際は，扱う計算機において計算時間の比率を計測し，最短の計算時間で実行できるように枝の本数を設定する [2, §4.2] 。本研究では，比率を 1 : 1 として実験した。実装は，まず optimal tree を作成し，その木の通りに同種写像計算と楕円曲線上の加算を繰り返す。

一般に ℓ 倍算と同種写像計算の計算時間比を $\alpha : \beta$ とするとき，optimal tree を生成するアルゴリズムは次の通りである。optimal tree は $e-1$ 次行列 $S_e = (s_{ij})$ として表し，その (i,j)-成分は E_i 上の点 $\ell^j P_i$ に対する作業を次のように表すこととする：

- $s_{ij} = 0$: 計算無し
- $s_{ij} = 1$: ℓ 倍算 $\ell(\ell^j P_i)$ を計算
- $s_{ij} = 2$: Vélu の公式 $\phi_i(\ell^j P_i) = \ell^j P_{i+1}$ を計算
- $s_{ij} = 3$: ℓ 倍算の結果を受けて ℓ 倍算と Vélu の公式を計算
- $s_{ij} = 4$: Vélu の公式の結果を受けて ℓ 倍算と Vélu の公式を計算

アルゴリズム 1 Optimal tree creation

Input: e, ℓ 倍算の重み α, 同種写像計算の重み β
Output: optimal tree S_e

$\mathrm{cost}_1 \leftarrow 0$, $\mathrm{cost}_2 \leftarrow \alpha + \beta$
$S_2 \leftarrow (4)$
for $n = 3$ **to** e **do**
  $\mathrm{cost}_k + \mathrm{cost}_{n-k} + (n-k) \times \alpha + k \times \beta$ を最小とする k ($1 \le k \le n-1$) を検索
  $\mathrm{cost}_n \leftarrow \mathrm{cost}_k + \mathrm{cost}_{n-k} + (n-k) \times \alpha + k \times \beta$
  $k_0 \leftarrow k$, $k_1 \leftarrow n - k_0$
  $S_n = \left(s_{ij}^{(n)}\right) \leftarrow n-1$ 次ゼロ行列
  for $j = 0$ **to** $k_1 - 1$ **do** $s_{0j}^{(n)} \leftarrow 1$
  for $i = 0$ **to** $k_0 - 1$ **do** $s_{i0}^{(n)} \leftarrow 2$
  S_{k_0} を k_1 列右シフトして S_n に上書き
  S_{k_1} を k_0 行下シフトして S_n に上書き
  $s_{0k_1}^{(n)} \leftarrow 3$
  if $k_1 > 1$ **then** $s_{k_0 0}^{(n)} \leftarrow 4$
  $s_{00}^{(n)} \leftarrow 4$
end for
return S_e

---

4 ℓ^e-同種写像計算の実装実験

本研究での実験環境は次の通りである。実装には Java の BigInteger クラスを用いた。

表 1: 実験環境

OS	Windows 10 Pro バージョン 22H2
CPU	11th Gen Intel® Core™ i9-11900K @ 3.50GHz 3.50GHz
メモリ	32.0GB (31.9GB 使用可能)
システムの種類	64 ビットオペレーティングシステム, x64 ベースプロセッサ
言語	Java 19.0.1
コンパイラ	javac 19.0.1

4.1 Optimal method の実装

Optimal method を実装し，パラメータを取り換えて実行時間を計測する。実験条件は次の通りである。$p = f \times \ell^e - 1$ としたときの ℓ^e のビット数を固定し，素数 ℓ は 2, 3, 5, 7, 11, 13 で比較する。また， f は ℓ^e のビット数と同じになるように設定する。入力する超特異楕円曲線 E_0，核 $P_0 \in E[\ell^e]$，点 R_0 は同じものを使用する。100 回実行した平均時間を実験結果とする。

Optimal method に対する simple method の実行時間の比率を測定した結果を表 2 に示す。

表 2: Optimal method に対する simple method の実行時間の比率

素数	ℓ^e のビット数			
ℓ	128	256	512	1024
2	6.0	10.3	18.0	32.3
3	6.3	10.9	19.2	34.4
5	4.3	7.3	12.8	22.7
7	3.2	5.6	9.6	17.0
11	2.5	4.1	7.0	12.2
13	2.5	3.7	6.4	11.1

表 2 より，ℓ^e のビット数を大きくするほど simple method に比べ速度の向上が大きくなる結果となった。e が大きいほど $O(e^2)$ と $O(e \log e)$ の差が大きくなることが影響すると考えられる。また，ℓ を大きくすると，simple method に比べ速度の向上割合が小さくなることが判明した。ℓ を大きくすると削減できる枝の本数が減少することに加え，一つ当たりの ℓ 倍算と同種写像の計算コストが増加することが原因であると考えられる。

次に，表 3 にそれぞれの ℓ^e のビット数に対して パラメータ毎に 1 回の同種写像計算にかかる計算時間を示す。

表 3: Optimal method の計算時間 [秒]

素数	ℓ^e のビット数			
ℓ	128	256	512	1024
2	0.02	0.12	0.66	4.51
3	0.02	0.10	0.59	3.96
5	0.02	0.10	0.60	4.09
7	0.02	0.11	0.67	4.54
11	0.03	0.13	0.77	5.18
13	0.03	0.14	0.79	5.36

Optimal method は $\ell = 2$ を除いて ℓ が小さいほど高速に動作することが判明した。

4.2 $\ell = 2, 3$ の場合に特化した Vélu の公式

高島 [5, p.12] では $\ell = 2$ の場合に特化した Vélu の公式が示されている。本研究では，$\ell = 3$ の場合にもそれに特化した Vélu の公式を用いることにより高速化を図った。

$\ell = 2$ のとき

楕円曲線 $E : y^2 = x^3 + ax + b$ 上の 2-等分点 $Q = (x_Q, 0)$ に対し，同種写像

$$\phi \colon E \longrightarrow E',$$
$$E' = E/\langle Q \rangle : Y^2 = X^3 + a'X + b'$$

は次式で与えられる。

$$t_Q := 3x_Q^2 + a$$

として，

$$a' := a - 5t_Q, \quad b' := b - 7x_Q t_Q,$$
$$X = x + \frac{t_Q}{x - x_Q}, \quad Y = y\left\{1 - \frac{t_Q}{(x - x_Q)^2}\right\}.$$

$\ell = 3$ のとき

核が 3-等分点のときも次のように Vélu の公式を簡潔に記述できる。

楕円曲線 $E : y^2 = x^3 + ax + b$ 上の 3-等分点 $Q = (x_Q, y_Q)$ に対し，同種写像

$$\phi \colon E \longrightarrow E',$$
$$E' = E/\langle Q \rangle : Y^2 = X^3 + a'X + b'$$

は次式で与えられる。

$$g_Q^x := 3x_Q^2 + a, \quad g_Q^y := -2y_Q,$$
$$t_Q := 2g_Q^x = 2(3x_Q^2 + a), \quad u_Q := (g_Q^y)^2 = 4(y_Q)^2$$

として，

$$a' := a - 5t_Q, \quad b' := b - 7(u_Q + x_Q t_Q),$$
$$X = x + \frac{t_Q}{x - x_Q} + \frac{u_Q}{(x - x_Q)^2}, \quad Y = y\left\{1 - \frac{t_Q}{(x - x_Q)^2} - \frac{2u_Q}{(x - x_Q)^3}\right\}.$$

上記の $\ell = 2, 3$ のときに特化した Vélu の公式に加え，2 倍算，3 倍算も最適化し optimal method に組み込む。本手法を optimal method with simple formulas と呼ぶことにする。Optimal method with simple formulas に対する optimal method の実行時間の比率を表 4 に示す。参考として，optimal method with simple formulas に対する simple method の実行時間の比率を表 5 に示す。

表 4: Optimal method with simple formulas に対する optimal method の実行時間の比率

素数	ℓ^e のビット数			
ℓ	128	256	512	1024
2	1.4	1.5	1.5	1.5
3	1.4	1.4	1.4	1.4

表 5: Optimal method with simple formulas に対する simple method の実行時間の比率

素数	ℓ^e のビット数			
ℓ	128	256	512	1024
2	8.6	15.0	26.5	48.0
3	9.0	15.3	27.0	48.4

Vélu の公式と楕円曲線上の加算を最適化させることにより，optimal method よりさらに 1.4 倍〜1.5 倍高速化した。次に，表 6 に，それぞれの ℓ^e のビット数に対して 1 回の同種写像計算にかかる計算時間を示す。

表 6: Optimal method with simple formulas の実行時間

ℓ^e の	実行時間 [秒]	
ビット数	$\ell = 2$	$\ell = 3$
128	0.016	0.014
256	0.079	0.071
512	0.451	0.417
1024	3.034	2.616

表 3 と 表 6 を比較した結果，optimal method with simple formulas は，どの ℓ についての optimal method の実行時間よりも短い時間で動作することを確認した。例えば SIDH 鍵交換では，小さな素数 ℓ を 2 つ用意する必要があるが，これらは 2 と 3 が適切であることが明らかになった。

5 SIDH 鍵交換, SIIBE の実装と実行速度

SIDH 鍵交換と SIIBE を実装し，実用的な鍵サイズでの実行時間を測定する。同種写像計算は，本研究で最も高速に動作した手法である optimal method with simple formulas を使用する。

5.1 SIDH 鍵交換の実装と実験

本節では 2.1 節で述べた SIDH を実装し，実行時間を測定する。実験条件は次の通りである。$p = f \times \ell_A^{e_A} \times \ell_B^{e_B} - 1$ に対して，$\ell_A = 2, \ell_B = 3$ とする。p は 128 ビット，256 ビット，512 ビット，1024 ビットでそれぞれ実験する。f は小さいビット数が望ましいため，ランダムに 20 ビット以下の正の整数を選び，残りの p のビット数を $\ell_A^{e_A}$ のビット数と $\ell_B^{e_B}$ のビット数が同程度になるように e_A と e_B を選択する。曲線は CM 法を用いて超特異楕円曲線を生成する。以上の条件の下，100 回 SIDH を実行した平均実行時間は次のようになった。

表 7: SIDH の平均実行時間

p のビット数	平均実行時間 [秒]
128	0.04
256	0.21
512	1.10
1024	7.97

p のビット数を 512 としたとしても，平均 1.1 秒で動作することを確認できた。

5.2 SIDH 鍵交換の数値例

p のビット数が 254 ビットのときの例を示す。記号は節 2.1 に準じている。

$$
\begin{aligned}
p &= 821567 \times 2^{118} \times 3^{74} - 1 \quad (2^{118}\text{のビット数}: 119,\ 3^{74}\text{のビット数}: 118) \\
&= 553548048688742963388729384064886141770808375206640410781348550381036235653 11 \\
E : y^2 &= x^3 + 40417794031241549707748494709499623049932040094453109358638148123059788634987\ x \\
&\quad + 45396797643785798584789975941829286758981639236523419931803717094741066945095 \\
&\quad (\mathrm{mod}\ 553548048688742963388729384064886141770808375206640410781348550381036235653 11) \\
P_A &= (\ 26102265900913132216324954701666615341828028826375526228233080646634393197343 \\
&\quad + 91676075728501950592535111010981332168554827340082595613857651659093215525 16\ i, \\
&\quad 47683926273775638456151675993898918279459482399291225844124528320515766142 85 \\
&\quad + 39583001829215754248811452187334769916944203611643454264261224649704078386749\ i\ ) \\
Q_A &= (\ 16419309770954556293126497788576157050064038769134975924381700696754829741385 \\
&\quad + 22575994599498173669315382478583702904999418550264451578345692007955594919 64\ i, \\
&\quad 48893904391053942864485084059857953532564607734098591914126203477456915075 86 \\
&\quad + 28625051516602030813594217943750193656409030131993570320006564372947737595941\ i\ ) \\
P_B &= (\ 61019966503296581091640873448070848684437448073456986402348474145012932132 52 \\
&\quad + 46991952733872132041913727733854989220878280421923551061526907057859451374442\ i, \\
&\quad 86209396990352195004015684449894036970277637897134830741852144355929313836 03 \\
&\quad + 77879744271798944049109643417177675947851124844051212763782156406172703636 45\ i\ ) \\
Q_B &= (\ 24101128355261807044712405767527199357425902865986973450467502226805766884768 \\
&\quad + 49165029885587917017542338015217591855693354201064484145172188061353199582513\ i, \\
&\quad 23986445707204726408489958199874465446190309807556707785450896728275365883 44 \\
&\quad + 31549824181238006877198296939319713141232493754640570983622159128396413813 91\ i\ ) \\
m_A &= 26215177061922878925482751457375001 1 \\
n_A &= 31858420855773481567649497150524263 1 \\
R_A &= (\ 53782868188384247654743708822445587164946676759377500736402535113619117773936 \\
&\quad + 31704242681374499664775643152283108860078619526920402922266435709666778532294\ i, \\
&\quad 41168751817644237151697513287953031686349884858796618423458388111096810056270 \\
&\quad + 51986863178962236537624390502832048946085803427391647593144777801203705754732\ i\ ) \\
m_B &= 13527321919661917873765551433476288 3 \\
n_B &= 16245179765278103942198543282995705 6 \\
R_B &= (\ 21631292719515335583655509038921161920274371981333374384015976459639655943229 \\
&\quad + 36642880056717294225714605777870396070268220187561760807501486127748745021695\ i, \\
&\quad 19599098566888746669505932500239841044752723388450140932596459162144142167373 \\
&\quad + 38047301902161696783172199552400250507381553142095232069780204615407420987455\ i\ ) \\
\varphi(P_B) &= (\ 36870835972505809056661128198724688404972144151309736682072564787786079672265 \\
&\quad + 51331115065573400473206687229385907015421818586125459254643049020725961020 28\ i, \\
&\quad 14113914106205049002557042656637100127393754113809433279694884745214534024260 \\
&\quad + 19550056221053948106817889373171760056678025692416547525116422050206616688092\ i\ )
\end{aligned}
$$

$$
\begin{aligned}
\varphi(Q_B) &= (\ 405929076835034987643738722135229055971505592102164768015717763207537884429 41 \\
&\quad + 704721870341857928287221946793660771042657599061346656639981605251538129481 9\ i, \\
&\quad 526614682362999437072876759115273527771036051165718485783487693743568537258 66 \\
&\quad + 292438598881129447929448798271293207662617269346300645160658332106789573256 3\ i\ ) \\
\varphi(R_B) &= (\ 458384268745321096745238221827908005663099620169751667081459274061878403450 6 \\
&\quad + 242616270285882645301530701105522172815136410196687581926450102181614738934 14\ i, \\
&\quad 511525168535529486706849433060656626028369236708331759001358445598219117574 80 \\
&\quad + 160280251062450302736253868842404473212215568313407184409519006015681358577 33\ i\ ) \\
\psi(P_A) &= (\ 185986229571144236137406432432113746117147082436991756037646000521516105176 04 \\
&\quad + 521625298611729252670567376098773542256509815481032988647361513148469814877 8\ i, \\
&\quad 122043108408496798006252185847039715025881348464349359115047718282903907846 97 \\
&\quad + 324159961523468206934996676286627250786069193143363551783580568879568233429 22\ i\ ) \\
\psi(Q_A) &= (\ 161172671261039773722807072867001679390384497899047340233005897894374588579 36 \\
&\quad + 547964995940416611617687536574245433534577978972121311199241607093470611909 26\ i, \\
&\quad 529348640641497473056207884917575694659624978165865909911640565738068909358 63 \\
&\quad + 230077409166315646144615625315417153805947340567823162707837516392957794293 61\ i\ ) \\
\psi(R_A) &= (\ 356886763864742947289014742263132126932821114870537870528694087133689990774 40 \\
&\quad + 470502638708025694479243201873638380450362091281886563619183643984289735657 87\ i, \\
&\quad 973359038471700217207465932720541772734454809293990695734923828045061698654 0 \\
&\quad + 440295808564693230181087824240890839588319396409195419270747036331854906663 25\ i\ ) \\
E_A : y^2 &= x^3 + (\ 409560929480668962946015690383266154290633539115882063274449022099248970896 89 \\
&\quad +377453000897998957787579515874095365645827426290844981312152967260834695216 38\ i\ )\ x \\
&\quad + (\ 470196718816348368303981619992112312322605984044232875789366153338667415842 35 \\
&\quad +481016416001680173714492026858127302463074983038959979274122240437244919611 23\ i\ ) \\
&\quad (\mathrm{mod}\ 553548048688742963388729384064886141770808375206640410781348550381036235653 11) \\
E_B : y^2 &= x^3 + (\ 126949863935853465700334847304698885057823373053407151921094291520486135851 3 \\
&\quad +466764499479688332257595270658763351962787854305180881556568237385551631448 60\ i\ )\ x \\
&\quad + (\ 444253486015433300087712494582133415373201899243903823371808530621621756175 11 \\
&\quad +100509299133569256117003137763283603235238657459759862156999904934111761044 48\ i\ ) \\
&\quad (\mathrm{mod}\ 553548048688742963388729384064886141770808375206640410781348550381036235653 11) \\
E_{AB} &= E_{BA} : \\
y^2 &= x^3 + (\ 514293300593127071614007271043413720757712438395928530348370924030723074851 77 \\
&\quad +792552364217806151521914204782906333367487196149793864763737999293776277429 4\ i\ )\ x \\
&\quad + (\ 204098551638744605792017626018806665833706417648148669642277733557886964066 39 \\
&\quad +134591236225483553312333238196873420546652845731856220449499529084023538733 95\ i\ ) \\
&\quad (\mathrm{mod}\ 553548048688742963388729384064886141770808375206640410781348550381036235653 11) \\
j(E_{AB}) &= j(E_{BA}) \\
&= 227317103157594147390494185558778885616708880070593472908028693430581990439 1 \\
&\quad +176254691190074016726990998030188421732871115118301492078246867242779321023 27\ i
\end{aligned}
$$

ただし $i = \sqrt{-1} \in \mathbb{F}_{p^2}$ とする。

5.3 SIIBE の実装と実験

本節では 2.2 節で述べた SIIBE を実装し，実行時間を測定する。実験条件は次の通りである。$p = f \times \ell^e - 1$ に対して，$\ell = 2$ とする。p は 128 ビット，256 ビット，512 ビット，1024 ビットでそれぞれ実験する。注意 2.2.1 より，f は大きくする必要があるため，f のビット数は 2^e のビット数と同程度になるよう設計する。曲線は CM 法を用いて超特異楕円曲線を生成し，ペアリングは Weil ペアリングを用いる。なお Weil ペアリング計算アルゴリズムも Java で実装した。以上の条件の下，100 回 SIIBE を実行した平均実行時間は次のようになった。

表 8: SIIBE の平均実行時間

p のビット数	平均実行時間 [秒]
128	0.02
256	0.10
512	0.46
1024	2.46

p のビット数を 512 としたとしても，平均 0.46 秒で動作することを確認できた。アルゴリズム中にペアリング計算を含むが，ℓ^e-同種写像計算の回数が少ない分，SIDH 鍵交換よりは短い実行時間で動作した。

5.4 SIIBE の数値例

p のビット数が 257 ビットのときの例を示す。記号は節 2.2 に準じている。なお本実験では，m-等分点に値をとるハッシュ関数は簡易的なものを用いた。

$$
\begin{aligned}
p &= 389914925368855257150786563197020999657 \times 2^{128} - 1 \\
&= 132681173702315141865438014550186471435326052993279445372984380168176079470591 \\
E_0 : y^2 &= x^3 + 23768282817421533350044451586087825948731877973852780463641192819242426934638\ x \\
&\quad + 15845521878281022233362967724058550632487918649235186975760795212828284623092 \\
&\quad \pmod{132681173702315141865438014550186471435326052993279445372984380168176079470591} \\
\phi\text{の核 } P &= (\ 81370727764711652912568091494244918775521733188745718659708080599275566126062 \\
&\quad + 11253472366766181110584613618677408346707728253500808569203107476495834986 3944\ i, \\
&\quad 49553030590094737682537350837927493634360832959191377217365039460118726974935 \\
&\quad + 70389499044337049845688120501007985593343159095606597984589527912548070956957\ i\ ) \\
E_1 : y^2 &= x^3 + (\ 88584009117305586086058882985827780040062253268660311292944211991843660284796 \\
&\quad +28502558832025109448464856585960442956088907003995086644063190989425492868334\ i\ )\ x \\
&\quad + (\ 47591816879788159289844701761563967006171704703542295545994480410002089988535 \\
&\quad +12145038303006381394587433161337661313450429744540999109332976459353882300022 2\ i\ ) \\
&\quad \pmod{132681173702315141865438014550186471435326052993279445372984380168176079470591}
\end{aligned}
$$

$$
\begin{aligned}
\hat{g_0} &= (\ 5124731171567137220108156104421422686149829425296559238349340777223195580854 3 \\
&\quad + 1112859870055685820600018075602117751972349931854320476533242203097589948824 43\ i, \\
&\quad 1448800132642475703660427321944487728260534412417424176800567974284842535116 2 \\
&\quad + 7324433074780376481127585429009897733747670743294493340648156173738813216479 2\ i\ ) \\
\hat{g_1} &= (\ 9969360144741838803505321461641949002701523167025989318806192469915133435840 2 \\
&\quad + 1026883793228424218121767561346435919845955017762995957353162427734298498980 92\ i, \\
&\quad 7006688367363959502588956083797326133996356017990016198090065477945332219822 4 \\
&\quad + 6712954171578634199207249740881359834454912076477363045629497405956068642164 6\ i\ ) \\
ID_X &= \mathsf{ksakaino@is.kochi-u.ac.jp} \\
h_{0X} &= (\ 2067850697498681974890156423883917108447428141554179503521574022410729686417 1\ , \\
&\quad 1073619320795465621688714418569863142699962322166654420410036217200566933597 98\ ) \\
h_{1X} &= (\ 1274056739888594761673632524969123992433140521542282910938344213501018526383 72 \\
&\quad + 3940310334579815945418553184551665915447694283604731594764531137576050004223 4\ i, \\
&\quad 4730409030147311366449282681886457870943362696818523918768588424158356638129 5 \\
&\quad + 1766059211215332492616276837633290149216848374629532888862194868724821179897 8\ i\ ) \\
M &= 3409605710093396734787869884346558091556479968076069198624812623205033524646 3 \\
&\quad + 7628236711277102924971896603875760057240897857805393140597146566353385254986 9\ i \\
c &= (\ 5912535268448410473705218553193364361561046670682421745346081702556794828526 4 \\
&\quad + 6755893327709685426884952092347261434766023254263202517905972577217985482860 3\ i, \\
&\quad 1142007527390500572264068510082095497192023079190327293442327539304224645588 61 \\
&\quad + 1127701695222898329527917290977082927833030578685827314595970103020976473139 9\ i\ ) \\
c_T &= 1091802135898884561301843058561142844936371188849068395639309566785416053763 62 \\
&\quad + 6373740556918602192730683190918327447492974289221972736763833594259564480731 8\ i \\
\mathrm{Enc}(M) &= (c, c_T) \\
\mathrm{Dec}(c, c_T) &= 3409605710093396734787869884346558091556479968076069198624812623205033524646 3 \\
&\quad + 7628236711277102924971896603875760057240897857805393140597146566353385254986 9\ i \\
&= M
\end{aligned}
$$

ただし $i = \sqrt{-1} \in \mathbb{F}_{p^2}$ とする。

6 まとめ

耐量子計算機暗号に用いる同種写像計算を実装し，動作速度の検証に加え，SIDH 鍵交換と SIIBE の実装を行った。同種写像計算は素朴な手法では計算回数の合計は $O(e^2)$ であるが，最適な計算手順を踏むことで $O(e \log e)$ まで削減することができる。本手法を実装することで，例えば ℓ^e のビット数が 256 のとき，素朴な手法に比べ 3.7 倍〜10.9 倍高速化した。また，同種写像計算ではパラメータとして小さな素数 ℓ が必要であるが，特に $\ell = 2, 3$ のときは Vélu の公式中の計算を簡略化することが可能である。楕円曲線上の 2 倍算，3 倍算も簡略化した関数を用いることで，更に 1.4 倍〜1.5 倍の高速化を実現できた。以上の結果より，ℓ^e-同種写像に用いる小さな素数 ℓ は 2, 3 が適切であった。SIDH と SIIBE は，素数 p のビット数が 256 のとき，SIDH は平均 0.21 秒，SIIBE は平均 0.10 秒で

動作した。また，p のビット数を 512 としても，SIDH は平均 1.10 秒，SIIBE は平均 0.46 秒で動作した。充分高速に動作し，実用上問題ない処理時間であることを確認できた。

参考文献

[1] Andreas Enge, Bilinear pairings on elliptic curves , HAL Id: hal-00767404 (2014).

[2] Luca De Feo, David Jao and Jérôme Plût, Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies, J. Math. Cryptol. 8 (2014), 209-247.

[3] Takeshi Koshiba and Katsuyuki Takashima, New Assumptions on Isogenous Pairing Groups with Applications to Attribute-Based Encryption. ICISC 2018, LNCS 11396 (2019), 3-19.

[4] Jacques Velu, Isogénies entre courbes elliptiques, C.R. Acad. Sc. Paris, Séries A. 273 (1971), 238-241. (https://aghitza.org/publications/translation-velu/ に Alex Ghitza による英訳がある)

[5] 高島克幸, 同種写像グラフの数理と耐量子計算機暗号への応用, 早稲田整数論セミナー, 2022 年 4 月 22 日.